Paper

Towards a Unified Architecture of Knowledge Management System for a Research Institute

Jarosław Sobieszek

Abstract—This paper presents some elements of architecture of planned knowledge management system dedicated to research institutions. Main contributions include social extension of the idea of adaptive hermeneutic agent and preliminary implementation of domain specific language for development of knowledge management systems. Work described here concentrated on practical verification of viability of proposed ideas and took form of a prototype software system, which can be used by a group of researchers to easily find and recommend relevant information.

Keywords—creativity support, knowledge management, model-driven software development, tagged collaborative filtering.

1. Introduction

One of the possible definitions of knowledge management [1] is:

"*Knowledge Management is the discipline of enabling individuals, teams and entire organisations to collectively and systematically create, share and apply knowledge, to better achieve their objectives*."

The tasks of knowledge management can be summarized by a checklist often used by journalists to verify that they present the whole picture of a situation. This list, known as *Five Ws and H*, consists of six interrogatives, which when applied to knowledge management [2] roughly correspond to:

- "when"' – time management,
- "what" – task management,
- "how" and "where" – information management,
- "who" – people management,
- "why" – goal management.

As can be seen, this list encompasses a wide range of different fields and potential techniques. Very often the solutions tend to be tailored to the needs of commercial entities, since they are created by large software corporations, which need to recoup their investment. The needs and expectations of research institutions are, to a significant degree, different and were only partially explored by theorists and especially by practitioners of knowledge management. One of the possible reasons for this disparity is a fact that processes of knowledge creation in academia are quite different to the ones at commercial institutions [3].

Our group at National Institute of Telecommunications, motivated to a significant degree by local practical need, decided to explore the topic of knowledge management in research institutions. Our main goal is creation of an integrated system which will merge traditional approaches to knowledge management with theories of creativity support [3]. Secondary requirements include architectural flexibility, which should simplify planned future deployments in other institutions, and low cost of proposed solution, to expand the group of potential users.

This paper presents results of an experiment conducted to investigate feasibility of proposed approach to development of knowledge management systems. We decided to limit the scope of this study to a small, but none the less useful part of the complete system, and focus our attention on the creativity support component. Section 2 presents three topics relevant to the presented application outlining, respectively, theory of knowledge creation, representation of preferences and interests, and integrated approach to development of information systems. Section 3 justifies some of design decisions and describes structure of the prototype. Section 4 summarizes main results presented in this article and details potential further enhancements and directions of future research.

2. Background

2.1. Creative Environment

Creative environment [3] is a comprehensive theory describing a place, where knowledge is created, shared and used. One of the most important aspects of this idea is identification of various knowledge creation processes and description of ways to support them.

The basis of this theory is formed by a model of knowledge creation called Nanatsudaki. The name is a Japanese phrase meaning *seven waterfalls*, and corresponds to the structure of this model. It is composed of seven so called knowledge creation spirals, which describe processes of knowledge creation typically encountered in both research and industrial institutions. The cyclic nature of the spirals

reflects the fact, that knowledge creation is a perpetual and self-propelled (positive feedback) endeavour.
One of these processes, called hermeneutic spiral, describes the activity of gathering sources, most often in a form of publications by other researchers, analyzing them and reflecting on them in search of new research ideas. It forms the very basis of a large part of scientific work. The hermeneutic spiral is also known as EAIR (enlightenment-analysis-hermeneutic immersion-reflection) spiral, an acronym derived from the four phases of this knowledge creation process.

- Enlightenment is a phase that starts with having an idea, which is considered to be worthy of further involvement and during which potential sources of information are explored and research materials are gathered.
- Analysis is a phase of rational study of relevant materials.
- Hermeneutic immersion is a phase during which concepts and ideas rationally explored in previous stage are absorbed into one's intuitive perception.
- Reflection is a phase during which new research ideas are intuitively considered and explored.

Another related concept is that of adaptive hermeneutic agent (AHA), a software system designed to support the knowledge creation process represented by the EAIR spiral. The original idea [3] described a system, which helped individual researcher to find relevant materials on the web and which was largely based on algorithmic analysis of document content. We decided to replace this mechanisms with framework for cooperation, motivated to some degree by growing importance of social web sites. This approach encourages collaboration and will hopefully lead to a more comprehensive realization of the idea of creative environment.

2.2. Tagged Collaborative Filtering

Knowledge representation is another important aspect of the architecture of knowledge management system. Here, we describe the representation of preferences and interests, since this information forms the basis of the proposed system.
Nowadays, Internet stores routinely use so called recommender systems [4] which propose goods the customer might be interested in buying. Generally, approaches to construction of these systems fall into two broadly defined categories.
First of them, called *content based*, concentrates on creating profiles which aim to explicitly describe both users and products. This technique relies on additional domain specific information, which could be hard to gather.
Alternative approach called *collaborative filtering* [5] or social information filtering relies only on past user behavior, predicting future interest based on preferences that were expressed by a preferably large group of users. These preferences could have been specified explicitly, taking form of ratings which quantify level of satisfaction, or could be extracted from more implicit sources, such as histories of purchases or page views. Generally such information is more readily available which partially explains relative popularity of solutions based on collaborative filtering. Additionally, it is a more versatile approach, since it does not depend on content being recommended.
Formally, let U and I denote, respectively, sets of users and items, with $|U| = n_U$ and $|I| = n_I$. Rating function $r : U \times I \rightarrow S$ is a mapping of user-item pairs into a rating scale S which is most often represented as a sequence of natural numbers, usually of length 5 or 10. Values of this function for a given sets of users and items can be tabularized to form a matrix $R = [r_{ui}]_{U \times I}$, where r_{ui} is a rating given by user u to item i. This formulation of the problem allows us to alternatively define collaborative filtering (or at least its most common form) to be an algorithm for estimation of missing entries of a matrix.
Tagging is another method commonly used for knowledge representation. The idea is to associate short phrases, known as tags, to provide additional information about some data. This is closely related to a concept of keywords used by librarians to index textual resources.
This two approaches can be merged to form what we have called *tagged collaborative filtering* which can be viewed as a multicriteria variant of collaborative filtering. Standard formulation of multicriteria analysis requires all values of the criteria to be specified, so a different name better reflects the fact that in this case they are optional. This approach is also quite similar to some of the methods used for content based recommendation systems, though one significant difference is that the tags can be used to not only describe content, but also, for example, preferences of the users.
Formally, we introduce another dimension into domain of the rating function which now becomes $r : U \times I \times T \rightarrow S$, where T is a set of tags. Both of the constituent ideas can now be expressed by imposing some limits on the dimensionality of sets used in its definition. Tagging is equivalent to reduction of rating scale S to a binary alternative, collaborative filtering is equivalent to reduction of tag set T to a single implied value which can be called quality or satisfaction.
Since collaborative tagging can be viewed as an approach to ontology construction, it should be possible to further extend this idea, and apply more sophisticated semantic structures to describe relations between tags, which could be then used for collaborative filtering.

2.3. Model-Driven Software Engineering

Software development is a complex process. One of the most common approaches to dealing with complexity is an

idea of splitting the problem into parts and dealing with them on an individual basis, also known as *divide and conquer*. When it is applied on a conceptual level, the parts are often called layers. This approach, when applied to software engineering, usually splits the process into three phases (analysis, design and development) corresponding to semantic, structural and technological aspects of the problem.

Duplication is probably one of the most pervasive problems in software development. It is usually considered harmful to the quality of the affected systems, though there are some specific cases when it is actually helpful, e.g., loop unrolling which repeats statements in the body of the loop to reduce the number of tests and jumps, often leads to a faster execution of the program and is one of the techniques used for code optimization. The advice of avoiding duplication was expressed by a pragmatic rule of software development [6], known as DRY (don't repeat yourself).

The risks of duplication of information were recognized, for example, in a field of relational databases, where a design technique called normalization [7] aims to minimize structural problems associated with having multiple sources of the same data. Designs which do not follow this practice are more susceptible to the occurrence of so called data anomalies, which can lead to a loss of data integrity.

Canonical layered approach to software development does not have any mechanisms which prevent duplication. It can be seen as one of the tasks of project manager. This arrangement can fail, especially since higher layers of this process often produce only design documents, which are often perceived only as a direction for future work. Additionally, since lower levels build upon previous steps, they tend to rephrase at least some of the work that was already done, which can introduce inconsistencies.

Model-driven software development (MDSD) is one of the possible techniques, which help to reduce duplication. It is a design philosophy emphasizing the role of models as a cornerstone of process of software creation.

Structure of model is determined by another model, called metamodel, which can be seen as a specification of vocabulary that can be used to define models. This class-instance relationship can be extended indefinitely, though in practice there is usually no need to go beyond three levels, with the most generic one defined in a recursive way. Model level is application specific, metamodel level provides a generalized view of a problem domain, and metametamodel level is associated with software development environment allowing it to access lower level constructs in a standardized way.

Individual models can be connected with transformations (see Fig. 1), which describe methods of converting one model into the other. Usually, conversion of models to/from their textual representation is treated separately using techniques, which facilitate text parsing and generation.

The process of software development can be seen from a global perspective as a directed graph, whose nodes are models and edges are model transformations. The sources, that is the nodes which are not a destination of any transformation, represent models which need to be specified by the developer. The transitions are another part which needs to be defined. The output of the process is represented by sinks, that is the nodes which are not a source of any transformation. They correspond for example to source code, documentation or user interface definitions.

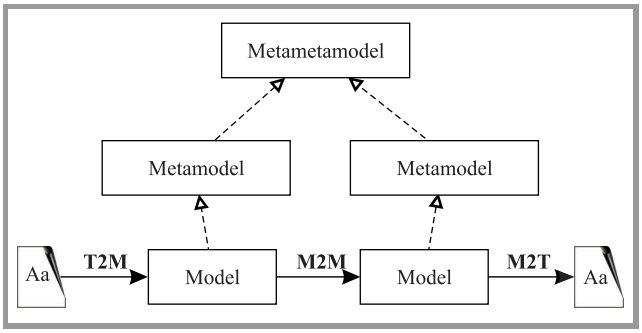


Fig. 1. Context of model transformations. Explanations: M2M – model to model, M2T – model to text, T2M – text to model.

One important consequence of imposing this kind of structure is that, the process of software development can be easily split into parts, which reflect certain perspectives, or ways of looking at the resulting system. For example, the process of development of data warehouse, can be split into several pieces: one that defines a transformation of domain specific model into a domain independent representation, the other one describes a way of implementing that representation in a specific runtime environment and yet another one specifies configuration information. This decomposition can reflect the structure of the development team, when the first transformation is defined by a business analyst, the second one by a software engineer, and the last one by maintenance staff.

Some of the other advantages of this approach include formalization of knowledge and greater potential for reusability. It forces the developer to formalize the approach used to solve the problem. From a point of view of future maintenance of the system it is a great advantage, since it documents all the decisions made by the developer and bridges the semantic gap that often arises between concept representations at different levels.

This technique is foremost a way to introduce static structure to the problem, so it won't be of much use in situations where the complexity is mostly of algorithmic nature. It will be of great help mostly in large heterogeneous information systems characterized by high structural and low algorithmic complexity, such as data warehouses or knowledge management systems.

We have decided to use probably the most popular approach to model-driven software development, namely model-driven architecture (MDA) [8]. It was developed under the auspices of Object Management Group (OMG), a widely known organization, which has, for example, standardized the Unified Modeling Language (UML).

The specific tool we have used is known as openArchitectureWare (OAW). It's a modular code generation framework, nicely integrated with Eclipse development environment and based on Eclipse Modeling Framework (EMF). One of the distinguishing features of this tool is its support for text to model transformations, which enables the developer to easily define domain specific languages (DSL).

3. Prototype

Primary goal of the work presented here was to explore the ideas and techniques described in Section 2. This examination took a form of a prototype software system, whose primary function is the ability to catalogue and search for various objects related to the field of research. On one hand, it can be viewed as a greatly simplified knowledge management system dedicated to research institutions, on the other hand, it is a social incarnation of an adaptive hermeneutic agent.

Social aspect of the system is emphasized by its approach to editing the data. It mimics wiki-like systems in that regard, allowing any user to add, edit and delete content from the database. With this freedom comes the disadvantage of increase in maintenance work, since information stored in the system needs to be protected from willful destruction. On the other hand, it lowers barriers to participation expanding the potential group of contributors. Wikipedia is a proof that this approach is both feasible and has a lot of potential.

Since semantic profile information needs to be stored on a per-user basis, to fully use the system one has to create an user account. The need to do this can be viewed as cumbersome, and potentially discourage some of the likely users. Therefore, we decided to make the registration process optional, and allow users to use the system without providing any additional information. Such passive users do not contribute to collaborative filtering, though hopefully if they find it useful, they will become more active participants. This reflects our philosophy that it is better to encourage than to force.

Another aspect that emphasizes this laissez-faire user experience is approach to ontology creation. Basically, there are two generic ways of building ontologies, known, respectively, as top-down and bottom-up approach. First of them is a more formalized process, where a group of experts progressively specializes the vocabulary used to describe the problem domain. Somewhat similar technique, known as mind mapping, is often utilized for brainstorming and note taking. The other approach starts with a collection of items describing the problem domain. They are analyzed to extract the most specialized concepts, which are then repeatedly generalized. This approach is susceptible to automation, where first step can use keyword extraction algorithms, followed by a series of clusterizations, to form the final ontology.

Ours is basically a bottom-up approach, though with one crucial difference, when compared to automatic method described above. It replaces computer algorithms with a framework for cooperation, which should allow interested parties to form the ontology as a byproduct of their evaluation of source material. This approach is known as folksonomy, which is portmanteau made by combining folk and taxonomy, and is often used to describe the emergent process of ontology creation happening in a group of collaborating people.

As was already mentioned, we decided to investigate the feasibility of using model-driven approach to construction of knowledge management systems. Thus, the backbone of prototype presented here is formed by a definition of a metamodel (Fig. 2), which formalizes vocabulary used to

```
System:
      "system" ":"
      (options+=Option | classes+=Class)*;
Option:
      "option" name=ID "=" value=STRING;
Class:
      "class" name=ID ":"
     (options+=Option | attributes+=Attribute)+;
Attribute:
      name=ID ":" type=Type (options=TypeParams)?;
Enum Type:
      string="String" | m2o="ManyToOne" |
      m2m="ManyToMany";
TypeParams:
       "(" TypeParam ("," TypeParam)* ")";
TypeParam:
      ID | INT;
```

Fig. 2. Specification of model parser.

describe the structure of this system. It is a simple object-oriented representation, composed of classes, which besides having attributes for storing values, can also be connected to each other with one of the two relations, namely many-to-one and many-to-many. Additionally both system and classes definitions can be annotated with metadata, which are called options here, that have a textual form and were used to specify labels displayed in the user interface. While not very elaborate, this metamodel is sufficient to describe a wide range of practical applications.

Based on the metamodel definition, we constructed a simplified model (Fig. 3) of publications catalogue. It consists of four classes, which represent respectively person, publication, institution and journal, connected with some self-explanatory relations. Thorough description of this particular application was not our goal, but it is something, that can be easily achieved. Thanks to chosen approach, what needs to be done from a technical point of view is a simple change of model definition. It is also possible to completely change the focus, and create, for example, a social bookmarking application or a movie database.

Again, all that is strictly necessary is a change of model definition.

```
system:
        class Person:
              first_name:  String(100)
              last_name:  String(100)
              affiliation:  ManyToOne(Institution)
        class Publication:
              title:  String(200)
              authors:  ManyToMany(Person)
              journal:  ManyToOne(Journal)
        class Institution:
              name:  String(200)
        class Journal:
              name:  String(100)
```

Fig. 3. Model of the prototype system.

The prototype took a form of a web application developed using Django framework. This allows it to be used on a variety of platforms, including, for example, mobile phones. Basic functionality focuses on providing create-read-update-delete (CRUD) interface to a catalogue describing some objects. User interface (Fig. 4) follows a common three-pane design. The central one displays information about object or a list of objects, the left one allows browsing specific classes of objects, and the right one provides interface for searching the database.

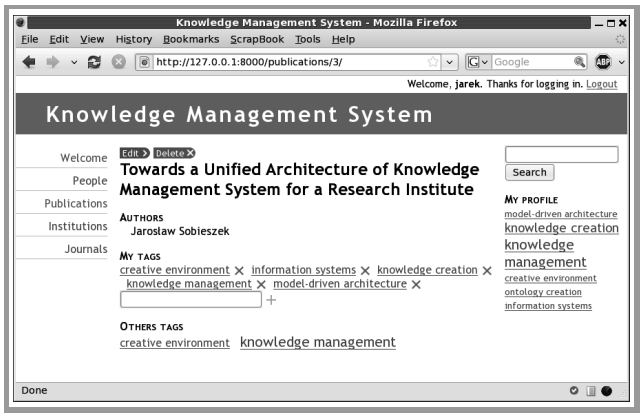


Fig. 4. User interface.

Functionality related to recommendation is at the moment limited to tagging. Every object can be annotated with keywords, which are then displayed in two separate lists. First of them shows tags of a logged in user, second one of all the other users aggregated to form a tag cloud. Keywords used by user to describe objects form a profile, also displayed as a tag cloud, which enables easy access to related content. Without logging in user cannot associate keywords with objects, and can only see a list of tags added by other people.

4. Conclusions and Future Work

In this paper we presented some elements of architecture of planned knowledge management system dedicated to research institutions. Main contributions include social extension of idea of adaptive hermeneutic agent and early stage of implementation of domain specific language for description of knowledge management systems. Work described here was preliminary, and its main goal was verification that proposed approach is viable direction of future efforts. The results of this feasibility study were encouraging, and we intend to build upon them in our forthcoming projects.

One of the more evident directions of future work is extension of adaptive hermeneutic agent component, which was only partially implemented. Especially, to fully utilize it, the profile needs to be directly editable and allow for more direct specification of preferences. Also metamodel, even though it is sufficient to describe a wide range of real world applications, needs to be extended, if it is to be used for construction of more comprehensive knowledge management applications. One simple, yet very powerful, addition would be introduction of processes [9], which are widely used for description of sequences of actions and, thus, well suited to support many of management tasks.

Other more long-term possibilities include addition of different algorithms for constructive manipulation of data gathered in presented system. For example, network structures could be analyzed, to compute impact factor of objects [10]. Similar approach is used by some search engines [11], and would extend scope of potential applications. Also interesting would be formalization of semantic structure of this system, built upon work done in fields of ontological engineering and semantic web [12], [13]. This would make the data amenable to more intricate automatic processing.

References

[1] R. Young, “Definition of knowledge management” [Online]. Available: http://www.knowledge-management-online.com/Definition-of- Knowledge-Management.html

[2] R. Young, “The future of knowledge management” [Online]. Available: http://knol.google.com/k/ron-young/the-future-of-knowledge-management/1emn5abyls393/4

[3] *Creative Environments: Issues of Creativity Support for the Knowledge Civilization Age*, A. P. Wierzbicki and Y. Nakamori, Eds., *Studies in Computational Intelligence*. Berlin-Heidelberg: Springer-Verlag, 2007, vol. 59.

[4] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions”, *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005.

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry”, *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[6] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*. Boston: Addison-Wesley, 1999.

[7] E. F. Codd, "A relational model of data for large shared data banks", *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[8] O. Pastor and J. C. Molina, *Model-driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling.* Berlin-Heidelberg: Springer-Verlag, 2007.

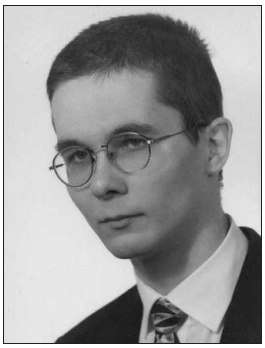
[9] M. Dumas, Wil M. van der Aalst, and A. H. ter Hofstede, *Process Aware Information Systems: Bridging People and Software Through Process Technology*. Hoboken: Wiley, 2005.

[10] J. E. Hirsch, "An index to quantify an individual's scientific research output", *Proc. Nat. Acad. Sci.*, vol. 102, pp. 165–169, 2005.

[11] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: the Science of Search Engine Rankings*. Princeton: Princeton University Press, 2006.

[12] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*. Cambridge: The MIT Press, 2004.

[13] *Towards the Semantic Web: Ontology-driven Knowledge Management*, J. Davies, D. Fensel, and F. van Harmelen, Eds. Chichester: Wiley, 2003.



Jarosław Sobieszek received his M.Sc. degree in computer science from Warsaw University of Technology, Poland, in 2002. Currently he is a researcher at National Institute of Telecommunications, where he prepares his Ph.D. thesis in the area of knowledge management. His research interests include machine learning, artificial intelligence, knowledge management and model-based approaches to software development.
e-mail: J.Sobieszek@itl.waw.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland